A novel Predictive Model for Determining Filtration Volume vs. Time for Nano Compounds with Multi-modal Particle Size Distribution

Modesto Torres
Electrical and Computer Engineering Department
US ARMY ARDEC, Picatinny Arsenal, New Jersey, USA
Stevens Institute of Technology
Hoboken, New Jersey, USA

***Abstract*— A novel predictive model for accurately determining filtrate mass correlation with respect with the input variables. The solution is based on training a neural network consisting of ten hidden layers using the Levenberg-Marcquard back propagation algorithm to recognize the correlation between the input variables, including the filtration time, and the filtrate mass.**

The main hurdle found was how to exactly organize the data in order to get the best Mean Square Error (MSE) and correlation (R) values to ensure a good prediction and a strong relationship between the input variables and the output variable which in this case is the Filtrate Mass.

This Model is proven to work with an estimated error of about 4.02% and 4.9% with a total sample size of 4974. Each sample consisted of a row containing 114 input variables, including the time for reaching the target Filtrate Mass, and one output variable which is the Filtrate Mass.

By using the proposed model, I demonstrated that with training sample of 3482 our model MSE is equal to 2.81284e-3 (very close to zero) and the correlation (R) is equal to 0.99999 (very strong, almost equal to 1). This is without the need to retrain the network and with only 1000 epochs for training validation and testing. Our validation sample size was 746 with a MSE equal to 3.40612e-3 (also very close to zero) and the correlation still 0.999999. Our testing sample size was 746 with the MSE equal 2.49244e-3 and R is still 0.999999.

The first stage for getting the model for this solution is to organize the data as a multidimensional array where the input variables are the known parameters of the compound including the time series of previously collected filtrate mass empirical data.

Many Models were tried but the model with the best MSE and R values was the model in which the complete PSD was inserted as variables. I decided to use the PSD particle size as a variable name and the volume in percent as the value for the variable and include it in the input array to the neural network. The output array of the neural network consisted of the empirical data of the filtrate mass over time. This model will work for n-modal compounds since all the information from the PSD is already taken into consideration with the model.

This model can be implemented in many ways. It could be made into a smart phone application using Java or the like, a computer application GUI or simply run from inside the Matlab® environment which was the one used for developing this model.

***IndexTerms*—nanoparticles, nanocompounds, PSD, filtration, filtrate, solution, multi-modal, volume, time, neural networks, model, segregation.**

I. Introduction

Nanomaterials and Nanoparticles are becoming part of everyday life in a broad spectrum of fields and applications in chemical industry, electronics, energy, bioindustry and medical technology, just to name a few. A Nanoparticle is defined as a nano-object with all three dimensions in the size range from 1 to 100 nano meters [1].

Numerous studies and analyses are trying to address the processing of Nanoparticles and nanocompounds as the technology tends to be a vital part of most known technologies to date. One of the fields with utmost importance is the development of separation processes, but very little contributions have been made to the studies of membrane filtration of nanoparticle suspensions [2].

The mechanisms and methods for filtration of Nanomaterials and Nanoparticles are difficult because their behavior is not fully understood. There is also a strong probability that these particles do not agglomerate on collision and their mean thermal velocity surpasses their capture velocity [3].

The method presented in this paper solves this problem algorithmically for any type of filter topology. Only the filter and nano-compound solution characteristics are needed to determine the volume of filtrate versus time.

II. SOLUTIION DETAILS

The model preprocessing includes the organization of the known data and empirical data for the compound. This model is strongly dependent of the compound PSD since all of the PSD particle sizes and volumetric percentages are fed to the network as variables. The elegancy of this model is the use of individual particle size as variables and the % volume in the PSD as the value for the variable. This makes the model more robust since the whole PSD (independent of particle size) is included in the model.

The model consists of an input (will depend on the total number of particle sizes in the PSD) that is fed into a neural network with 10 hidden layers and one output layer plus an output which consists of the Filtrate Mass variable. The neural network topology is shown in Figure 1. This figure shows an input of 114 since the parameter data plus the PSD values added up to 114 variables.

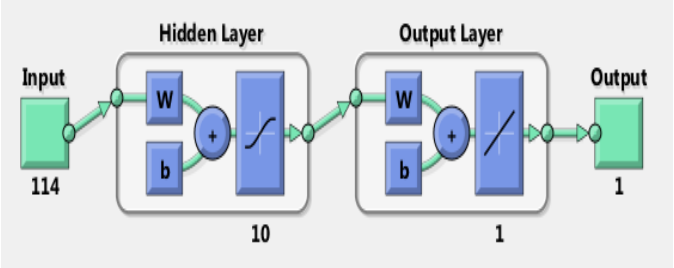

Fig. 1: Neural Network Topology

III. INPUT ORGANIZATION

The input variables to be fed to the Neural Network were organized into a single array of variable number columns and variable number of rows. I called this array nn_input. This array consists of all the known and simple parameters for the compound. The variable names are the column labels for each case in the following order:

1. Filter diameter
2. Filter area
3. Filtration temperature
4. Filtrate density
5. Viscosity n-butanol
6. Viscosity water
7. Volume fraction n-butanol (solids free)
8. Volume fraction water (solids free)
9. Viscosity saturated solution
10. Mass of dry cake from filtration
11. Solvent loss on drying of wet cake
12. Filtration pressure
13. Time(s)
14. PSD Particle Size 1 (um)
15. PSD Particle Size 2 (um)
16. .
17. .
18. .M. PSD Particle Size N (um)

Please note that the nn_input array is a variable size array of size M Columns and N rows (number of data points to be fed into the neural network) since it will depend on the total number of particle sizes in the PSD and the number of samples in our empirical or known data. Also note that the values for PSD Particle Size 1 (um) to PSD Particle Size N (um) columns are the Volume (%) in the PSD.

Other known variables could be included in the nn_input array if they are known would help on improving the performance of the network.

IV. THE OUTPUT VARIABLE

The output variable is as simple as that is an array of 1 column by M rows containing the Filtrate Mass (g) variable at specific times.

V. NEURAL NETWORK PERFORMANCE

The proposed model was tested with the empirical data from two types of filter examples. Several tools were used but the one found most suitable for data manipulation and neural network implementation was Matlab® Release 2012b using the Neural Network Toolbox. A portion of the data was used as training data (70 %), 15% of the data was used for validation and the other 15% of the data was used for testing the network. The **nn_input** array consisted of 114 variables (columns) and 4974 samples (rows). Figure 2 shows the percentages and actual number of samples for filter one data. This example was tested from 0 time to the Maximum time provided which was 2626 seconds.

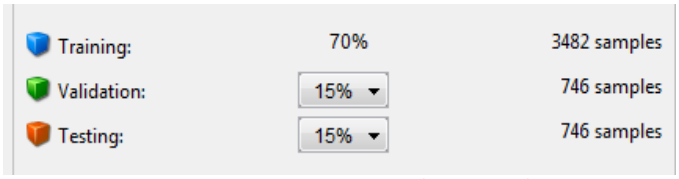

Fig. 2. Percentages and number of samples for Training Validation and Testing of the Neural Network for Example 1

Figure 3 is the explanation of the three kinds of samples that were used in the neural network training, evaluation and testing. The samples were selected randomly for each kind of samples

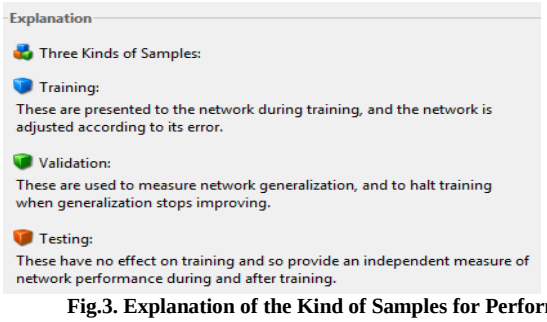

Fig.3. Explanation of the Kind of Samples for Performance Measurement

After the input (nn_input) and output (nn-output) data was organized. This information was fed to the neural network. I included the Matlab® 2012b script for finding the MSE and correlation between the inputs and outputs for Training, Validation and Testing.

The following is the Matlab® script. If the reader wants to evaluate the network with their own data, the data (nn_input and nn_output) shall be in the Matlab® workspace already and the following code can be copied and pasted into a new Matlab® script and run it. You must be aware that you must have the Neural Network Toolbox installed. I had to develop a tool to organize the data into the nn_input and nn_output from the empirical data taken at the laboratory. You must do the same with your own data.

VI. MATLAB® SCRIPT

This section describes the Matlab® code for Replicating the experiment with user data:

```
%Solve an Input-Output Fitting problem with a Neural Network
% This script assumes these variables are defined:
%
%   nn_input - input data.
%   nn_output - target data.

inputs = nn_input';
targets = nn_output';

% Create a Fitting Network
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,inputs,targets)
%figure, plotregression(targets,outputs)
%figure, ploterrhist(errors)
```

VII. NEURAL NETWORK PERFORMANCE

I included some of the plots from my multiple runs in the following sections. Many trials were performed to optimize the network for better performance by adding more neurons, more layers and varying the testing validation and training sample sizes. But it turned out that 1000 neuron network and a 10 neuron network will have almost the same performance while the 10 neuron network will have a solution within a reasonable time frame (around 1 minute with almost 5000 rows and 115 variables). So I decided to present the solution with a 10 neuron for this reason. Changing the sample size gave me almost the same performance but the optimum levels are the ones presented in this solution.

This neural network was run for 1000 epochs. This section includes the charts for performance at 22 epochs and 1000 epochs and shows a very fast network convergence into the final values in a very short time. The network convergence is very close to the final value during the first ten iterations while

the maximum performance is achieved very close to the four hundredth iteration.

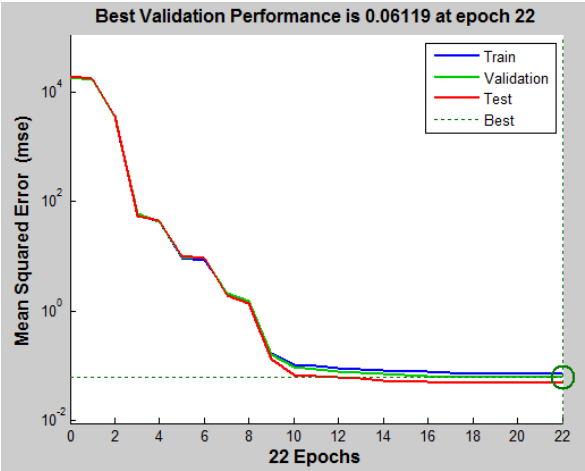

Fig. 4. Validation Performance at epoch 22

A. Prediction Error

The error histogram presented in the next figure shows that the error for the training, validation and test was between 4.02% and 4.9% which is an acceptable error.

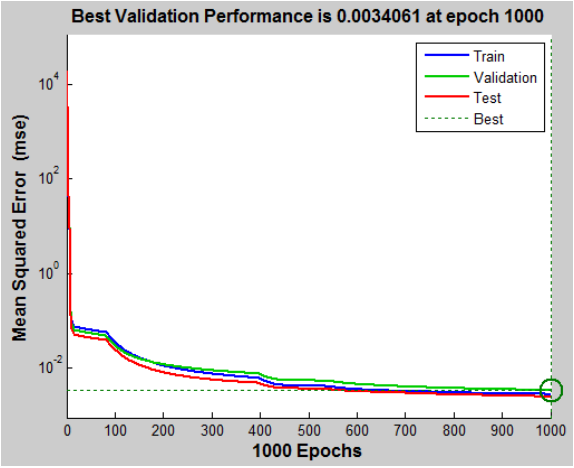

Fig. 5. Validation Performance at epoch 1000

B. Regression Plot

The following plots show a regression plot for the training, validation and test for output vs. target. There is another plot for the combined correlations for validation, training and test sets showing n almost perfect correlation between the outputs of the network vs. the target value of the Filtrate Mass variable.

VIII. CONCLUSIONS

A novel predictive model for accurately determining filtrate mass versus time was developed. The solution is based on training a neural network consisting of ten hidden layers using the Levenberg-Marcquard back propagation algorithm to recognize the correlation between the input variables, including the filtration time, and the filtrate mass.

The Network was trained with 3482 random samples, validated with 746 random samples and tested with 746 random samples from empirical data for two different types of filter topologies.

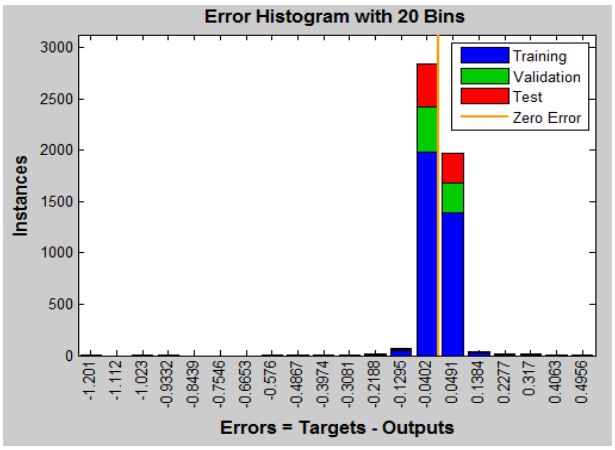

Fig. 6. Error Histogram with 20 Bins

The first stage for getting the model for this solution is to organize the data as a multidimensional array where the input variables are the known parameters of the compound including the time series of previously collected filtrate mass empirical data.

This Model is proven to work with an estimated error of about 4.02% and 4.9% with a total sample size of 4974. Each sample consisted of a row containing 114 input variables, including the time for reaching the target Filtrate Mass, and one output variable which is the Filtrate Mass.

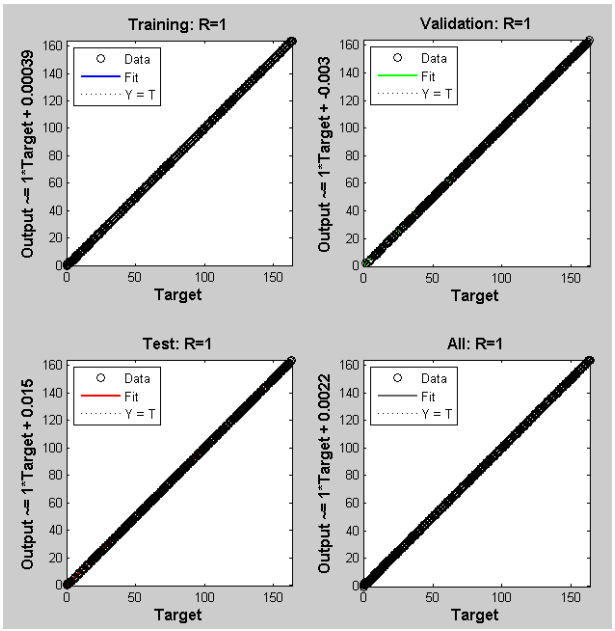

Fig. 7. Regression Analysis Results for Training and Validation

The Network was trained with 3482 random samples and validated with 746 random samples and tested with 746 random samples from empirical data for two different types of filter topologies.

This model demonstrated that with training sample of 3482 our model MSE is equal to 2.81284e-3(very close to zero) and the correlation (R) is equal to 0.99999(very strong, almost equal to 1). This is without the need to retrain the network and with only 1000 epochs for training validation and testing. Our validation sample size was 746 with a MSE equal to 3.40612e-3 (also very close to zero) and the correlation still 0.999999. Our testing sample size was 746 with the MSE equal 2.49244e-3 and R is still 0.999999. All this data can be seen graphically in Figures 6 (error histogram) and 7 (regression analysis), while the demonstration of a very fast convergence to the final values can be seen in figures 4 and 5.

The first stage for getting the model for this solution is to organize the data as a multidimensional array where the input variables are the known parameters of the compound including the time series of previously collected filtrate mass empirical data.

ACKNOWLEDGMENT

I would like to thank Mr. Patrick Calella for his support in the verification of this model.

This project was funded by Grant # 1301 form Icon Corporation Puerto Rico.

REFERENCES

[1] ISO/TS. Nanotechnologies—terminology and definitions for nano-objects—nanoparticle, nanofibre and nanoplate. Geneva, Switzerland: International Standards Organization; 2009

[2] Yasuhito Mukai and Aya Nishio, Yasuhito Mukai and Aya Nishio,“Characteristics of Filter Cake Exfoliation in Upward Ultrafiltration of Nanoparticle Suspensions”, Membranes 2011, 1, 59-69; doi:10.3390/

[3] Chan soo Kim, Li Bao, Kikuo Okuyama, Manabu Shimada and Hitoshi Ninuma,”Filtration Efficiency of a fibrous Filter for Nanoparticles”, Journal of Nanoparticle Research 92006) 8: 215-221.